
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS y NATURALES

U.N.C.

Cátedra de Sistemas de Control II

Año 2023

Informe de Trabajo Práctico de Laboratorio

"ARDUINO HAND CAR"

Britez Fabio

Diaz Mateo

Rodriguez Facundo Nicolas

Vega Guadalupe

Docente : Ing. Hugo Pailos

Índice

Introducción

Este trabajo se enfoca en la idea de control wireless, como así también en recrear algún autómata que nos ayude en determinadas circunstancias. En este caso se planteó la idea de un auto tomando referencias de ejemplo como los que trabajan en industrias,hogares y también los reconocidos “Rover” de la NASA. Todos estos disparadores fueron la base de este proyecto lo cuales aplicando conocimientos de Sistemas de control I , Teoría de las comunicaciones, se definió el “Arduino hand car”.

Objetivos

Desarrollar un sistema de control en Arduino que permita controlar nuestro arduino car de manera inalámbrica a una determinada distancia.

Marco teórico

IMU

Una unidad de medición inercial (IMU) es el nombre genérico para denominar a un dispositivo que es capaz de medir la velocidad, orientación y aceleración de un sistema. La mayoría parten de una combinación de acelerómetro y giroscopio, siendo este el IMU más habitual. El motivo es que ambos dispositivos combinan muy bien y compensan las limitaciones del otro.
En nuestro caso nos basamos en el uso del acelerómetro específicamente.
Este módulo se utilizan en mediciones de aceleración gravitacional estática, lo que le permite determinar el ángulo de desviación del objeto medido de la vertical, como desventaja , se ven influenciados por los movimientos del sensor y ruido por lo que no son fiables a corto plazo
Para poder ello es necesario que filtremos nuestra señal obtenida del sensor.
Existen varios filtros posibles como por ejemplo:

- Filtro de Kalman
- Filtro Complementario

Filtro de Kalman

El Filtro de Kalman es uno de los algoritmos de estimación más importantes y comunes. Es un método matemático que produce estimaciones de variables ocultas basadas en mediciones inexactas e inciertas, estima el estado de un sistema dinámico a partir de medidas ruidosas y variables no observables.
Además, el filtro de Kalman proporciona una predicción del estado futuro del sistema, basado en estimaciones pasadas. A grandes rasgos el filtro de Kalman realiza una estimación del valor futuro de la medición, y después compara el valor real mediante un análisis estadístico para compensar el error en futuras mediciones. Sin embargo su realización implica cálculos complejos que suponen una implementación y tiempo de cálculo excesivo para Arduino por lo que se suele utilizar una simplificación de Kalman, conocida como Filtro Complementario.

Filtro Complementario

El filtro complementario puede considerarse una simplificación del filtro de Kalman que prescinde por completo del análisis estadístico, puede expresarse como:

$$\theta = A \cdot (\theta_{prev} + \theta_{gyro}) + B \cdot \theta_{accel}$$

Donde A y B son dos constantes que, inicialmente, puede tomarse 0.98 y 0.02 respectivamente. Podemos calibrar el filtro simplemente variando los valores de A y B siempre que cumplamos la condición de que sumen 1 entre ellos.

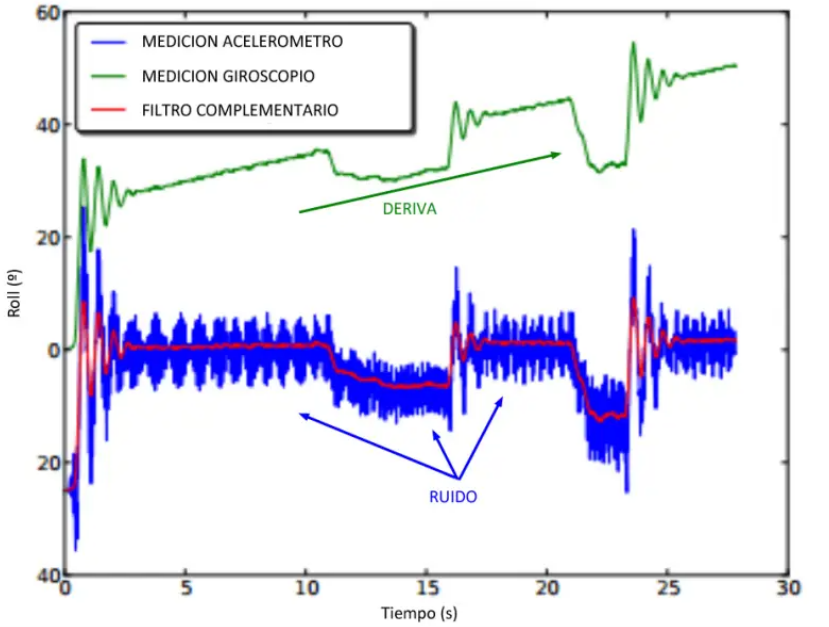

Fig 1.

El filtro complementario se comporta como un filtro de paso alto para la medición del giroscopio y un filtro de paso bajo para la señal del acelerómetro.Es decir, la señal del giroscopio manda a corto plazo, y la del acelerómetro y medio y largo, que es exactamente lo que queremos para compensar sus ventajas y defectos.

Planificación

Actividades	Planeamiento y Obtención de componentes		Diseño y programación					Montaje y Presentación		
Fechas	3-abr	12-abr	24-abr	5-may	8-may	17-may	29-may	7-jun	12-jun	19-may
Grupo	Investigación y Evaluación de posibles proyectos							Comienzo de armado de informe	Finalización del informe	Presentación del trabajo virtual
	Elección final de proyecto					Investigación para el analisis del modelo matemático del sistema		Montaje final con guante incluido	Realización del video	
Mateo y Facundo		Obtención de Transmisor y Receptor NRF24L01	Programación del movimiento del auto utilizando un driver L298N con los cuatro motores	Código para la transmición serial	Se logró la conexión y comunicación entre el auto y una pc	Mediones para estudio del sistema con 12V	Mediciones del sistema con los 5V del power bank			
		Prueba de comunicación serial con transmior		Prueba de comunicación serial, se envió una letra y con el receptor se	Correción del código final para el acelerómetro	Montaje del power bank de 5V para el auto-arduino	Modelo matemático para sistema de segundo grado con los parámetros recalculados			
Fabio y Guada		Obtención y prueba de los 4 motores	Obteción del acelerómetro	Configuración del acelerómetro	Diagrama de Bloques con descripción de los componentes	Pruebas de velocidad para el grafico v-t	Montaje del power bank de 5V DC el cual nos permite alimentar el driver del motor			
		Armado de hadware del auto-arduino	Investigación del filtro de kalman y complementario	Código de prueba de ir hacia adelante con el acelerómetro	Implementación del filtro complementario	Armado del gráfico v-t	Fabricación del guante para maniobrar con el acelerómetro			

Fig 2.

Diagrama de bloques

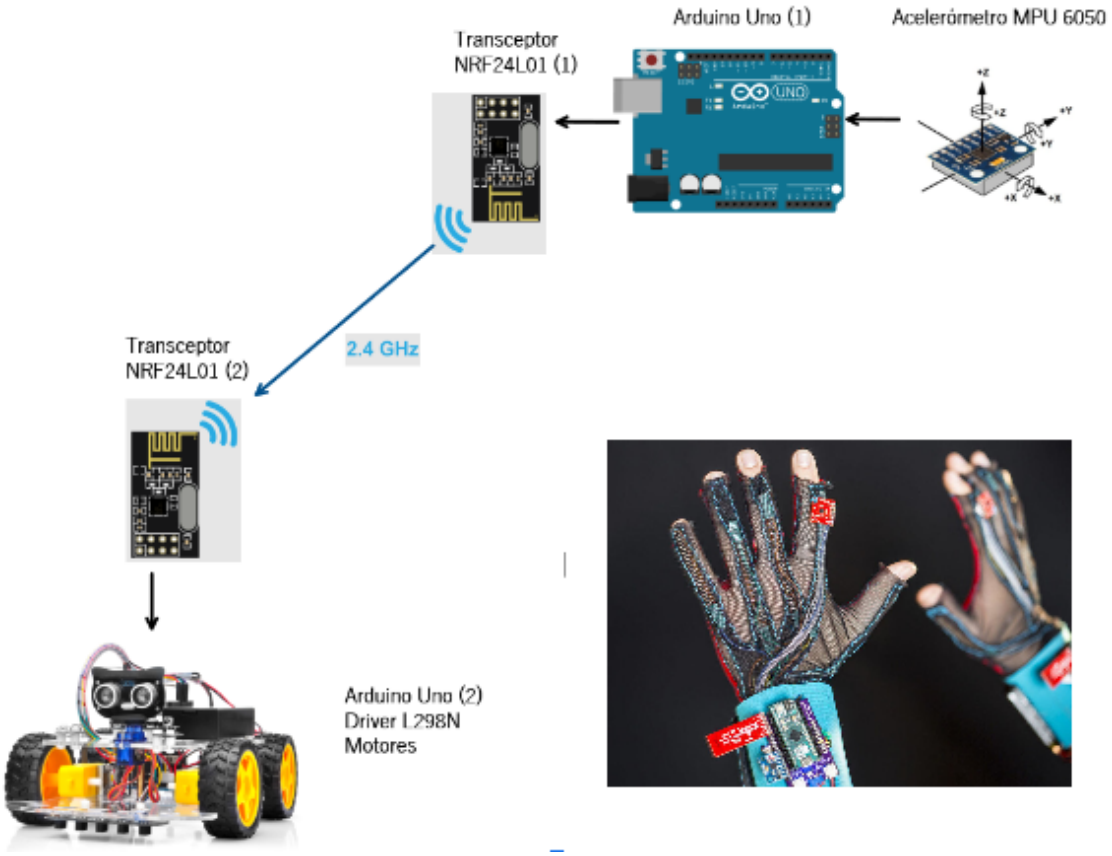

Fig 3.

Diseño electrónico

Se realizará una listas de los componentes a utilizar y una breve descripción de los mismos:

Acelerómetro MPU6050:

Mide el ángulo de la mano y envía la información procesada a los pines del arduino para su posterior procesamiento.Según la información recibida y la calibración del mismo podremos dirigir movimientos varios.

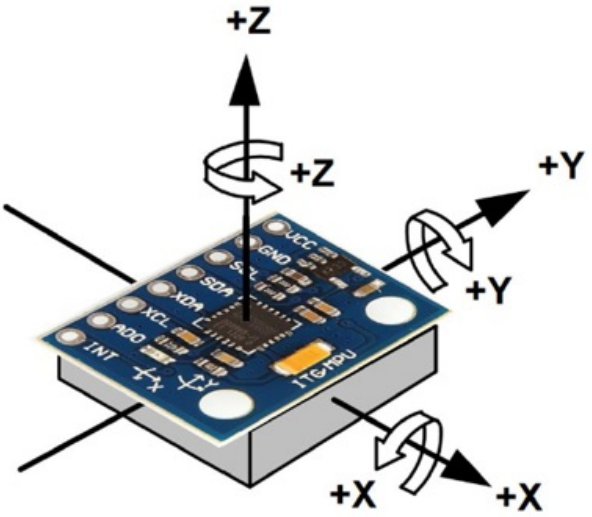

Fig 4.

Arduino Uno (1)

Obtiene los datos del acelerómetro,procesa y los envía al transceptor RF.

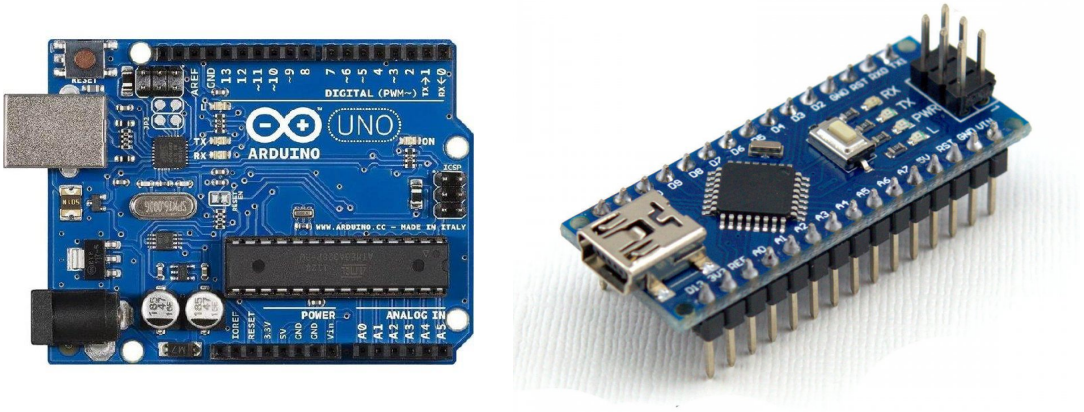

Fig 5.

Transceptor NRF24L01 (1):

Cumple la función de transmisor. Envía la información al segundo transceptor

Transceptor NRF24L01 (2):

Cumple la función de receptor. Una vez recibe la información se la envía a los pines del arduino para su procesamiento.

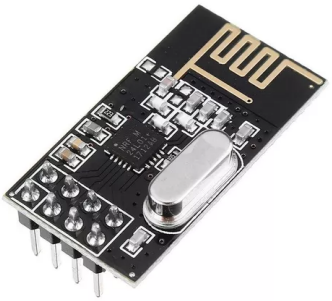

Fig 6.

Arduino Uno (2):

Procesa la información recibida y según los datos decide la velocidad y la dirección del motor. Luego envía las instrucciones al driver.

Driver L298N:

Controla ambos motores con las instrucciones recibidas por el arduino.

Fig 7.

Estructura auto en acrilico:

Fig 8.

Esquema de conexionado

Transmisor

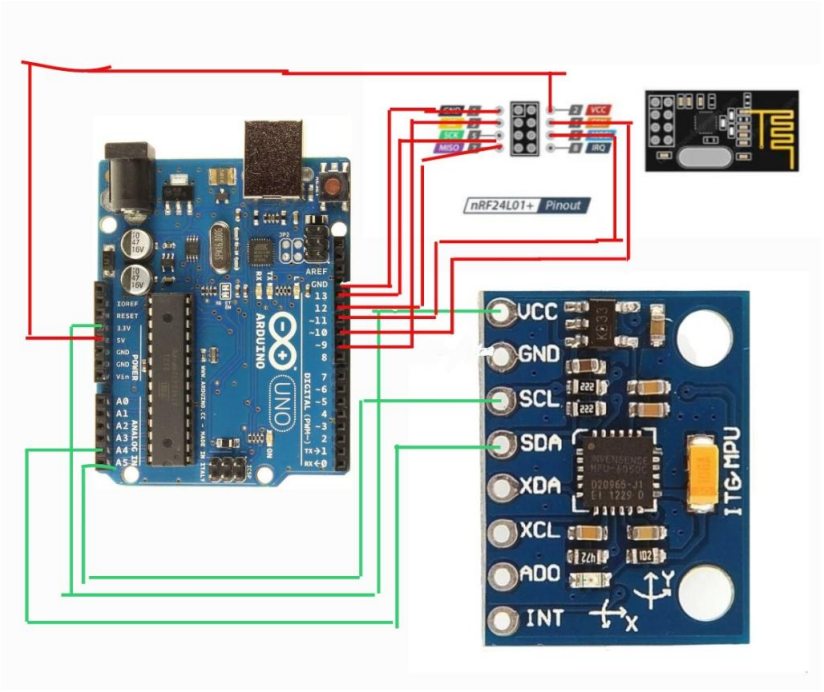

Fig 9.

Receptor

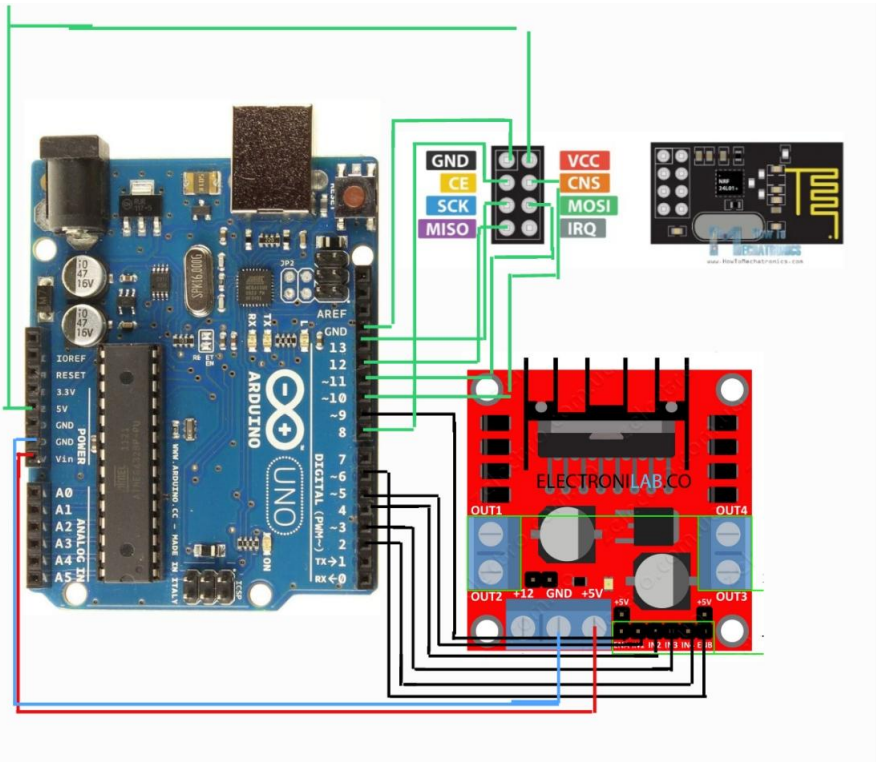

Fig 10.

Diseño del software

Para el mismo tenemos que tener en cuenta dos partes, la comunicación y la parte de control de los motores. Por el lado de la comunicación tenemos al receptor y transmisor. Encargados de llevar los datos a través de RF y así comunicar los datos proporcionados por el acelerómetro, codificarlos y darle las órdenes al programa encargado del control de los motores.

Para poder lograr la transmisión se hizo uso de las librerías “<SPI.h>“ y <RF24.h> las cuales contienen el programa con los protocolos de comunicación y así poder entablar contacto entre el transmisor y receptor. Con la ayuda de estas librerías y estas líneas de código, se logra transmitir las órdenes desde el acelerómetro y el microcontrolador encargado del movimiento del auto.

Transmisor:

```
#include <SPI.h>          // protocolo de SPI (Serial Protocol Interface)
#include <RF24.h>         // radiofrecuencia nRF24L01
#include "I2Cdev.h"       // comunicaciones I2C
#include "MPU6050.h"      // sensor MPU6050
#include "Wire.h"         // comunicaciones I2C

MPU6050 sensor;
int ax, ay, az;           // aceleración en los ejes x, y, z
```

```
int gx, gy, gz;          // giroscopio en los ejes x, y, z

long tiempo_prev;        // tiempo previo
float dt;                // intervalo de tiempo
float ang_x, ang_y;      // angulos de rotación en los ejes x, y
float ang_x_prev, ang_y_prev; // ángulos de rotación anteriores

RF24 radio(9, 10);               // Inicialización del módulo RF24
const byte address[6] = "00001"; // Dirección del módulo de radio

// Estructura para almacenar los datos a transmitir
struct Data {
 int valueX;
 int valueY;
};

void setup() {
 ang_x_prev = 0;
 ang_y_prev = 0;
 Serial.begin(9600);     // Comunicación serial a 9600 baudios
 Wire.begin();           // Inicio de comunicación I2C
 sensor.initialize();    // Inicio del sensor MPU6050

 // Comprobación de la conexión con el sensor
 if (sensor.testConnection())
 {
   Serial.println("Sensor iniciado correctamente");
 } else {
   Serial.println("Error al iniciar el sensor");
 }

 // Inicializaciones del módulo de radio
 radio.begin();
 radio.openWritingPipe(address);
 radio.setPALevel(RF24_PA_LOW);
 radio.stopListening();
}

void loop() {
 // Obtención de datos de aceleración y velocidad angular
 sensor.getAcceleration(&ax, &ay, &az);
 sensor.getRotation(&gx, &gy, &gz);
  // Cálculo del intervalo de tiempo
```

```
 dt = (millis()-tiempo_prev)/1000.0;
 tiempo_prev=millis();
  // Cálculo de los ángulos con el acelerómetro
 float accel_ang_x = atan(ay/sqrt(pow(ax,2) + pow(az,2)))*(180.0/3.14);
 float accel_ang_y = atan(-ax/sqrt(pow(ay,2) +
pow(az,2)))*(180.0/3.14);
  // Cálculo del ángulo de rotación con el giroscopio y filtro
complementario
 ang_x = 0.98*(ang_x_prev+(gx/131)*dt) + 0.02*accel_ang_x;
 ang_y = 0.98*(ang_y_prev+(gy/131)*dt) + 0.02*accel_ang_y;
  ang_x_prev = ang_x;
 ang_y_prev = ang_y;

 // Enpaquetar los datos para la transmisión
 Data data;
 data.valueX = ang_x;
 data.valueY = ang_y;
 // Transmisión de los datos
 radio.write(&data, sizeof(data));

 delay(10);
}
```

Receptor:

Para poder controlar los motores, también se hizo uso de un microcontrolador arduino el cual mediante el siguiente código controlaba simultáneamente los 4 motores.

```
#include <SPI.h>
#include <RF24.h>

RF24 radio(8, 10);                  // Enable (CE) en el pin 8 y Chip
Select Not(CSN) en 10
const byte address[6] = "00001";    // Dirección del receptor
int angleX = 0;                     // Ángulo X para adelante/atrás
int angleY = 0;                     // Angulo Y para izquierda/derecha
int powerA, powerB;                // Variables para el PWM

const int N1 = 5; // Pines para la dirección del motor A
const int N2 = 4;
const int N3 = 3; // Pines para la dirección del motor B
const int N4 = 2;
```

```
struct Data {     // Estructura de datos para recibir los ángulos
 int valueX;
 int valueY;
};

void setup()
{
   pinMode(N1, OUTPUT);       // N1 - Motor A
   pinMode(N2, OUTPUT);       // N2 - Motor A
   pinMode(N3, OUTPUT);       // N3 - Motor B
   pinMode(N4, OUTPUT);       // N4 - Motor B

   pinMode(6, OUTPUT);        // ENB (Motores B)
   pinMode(9, OUTPUT);        // ENA (Motores A)

   Serial.begin(9600);
   radio.begin();
   radio.openReadingPipe(0, address);
   radio.setPALevel(RF24_PA_LOW);
   radio.startListening();
}

void loop(){
   if(radio.available())
   {
     Data data;
     radio.read(&data, sizeof(data));
     int angleX = data.valueX;     // velocidad de adelante/atrás
     int angleY = data.valueY;     // velocidad de giro
izquierda/derecha

     setDirectionAndVelocity(angleX, angleY);

     Serial.print(angleX);
     Serial.print(",");
     Serial.println(angleY);
     delay(10);
   }
}

 void setDirectionAndVelocity(int anguloX, int anguloY)
{
 // Asegurarse de que anguloX y anguloY están en el rango de 0 a 255
```

```
 anguloX = constrain(anguloX, -255, 255);
 anguloY = constrain(anguloY, -255, 255);

 int powerA = anguloX + anguloY;
 int powerB = anguloX - anguloY;

 // Asegurarse de que powerA y powerB están en el rango de 0 a 255
 powerA = constrain(powerA, 0, 255);
 powerB = constrain(powerB, 0, 255);

 // Definir la dirección de los motores
 bool dirA = (anguloX >= 0);
 bool dirB = (anguloY >= 0);

 // Establecer la dirección de los motores
 digitalWrite(N1, dirA);
 digitalWrite(N2, !dirA);
 digitalWrite(N3, dirB);
 digitalWrite(N4, !dirB);

 // Establecer la velocidad de los motores
 analogWrite(6, powerA);
 analogWrite(9, powerB);
}

void stop()
{
   digitalWrite(N1, LOW);
   digitalWrite(N2, LOW);
   digitalWrite(N3, LOW);
   digitalWrite(N4, LOW);
}
```

Análisis respuesta a nuestro sistema

Realizamos bajo un proceso de medición en el laboratorio, realizando medidas y calculando la velocidad, pudimos entonces estudiar nuestro sistema, y obtener así nuestra función de transferencia para el estudio del mismo.

Mediciones

En este experimento, establecimos de antemano ciertos valores para la señal de entrada, en este caso, una modulación de ancho de pulso (PWM). Para registrar los datos de posición y tiempo, dispusimos una cinta métrica a lo largo del suelo y ubicamos el automóvil sobre esta.

Después de preparar el escenario, encendimos el auto y procedimos a grabar varios videos, asegurándonos de enfocar constantemente la cinta métrica. Esta metodología nos permitió capturar con una cierta precisión las posiciones del automóvil y los tiempos correspondientes en cada instante. A partir de los datos de posición y tiempo capturados, realizamos un cálculo sencillo para determinar la velocidad del automóvil en cada momento del experimento.

Tiempo(s)	Posición(m)	Velocidad(m/s)
0	0	0
1,2	0,4	0,3333333333
1,876	0,6	0,2958579882
2,543	0,8	0,299850075
3,22	1	0,2954209749
3,57	1,1	0,2857142857
5,12	1,5	0,29296875
5,82	1,7	0,2857142857
6,52	1,9	0,2857142857

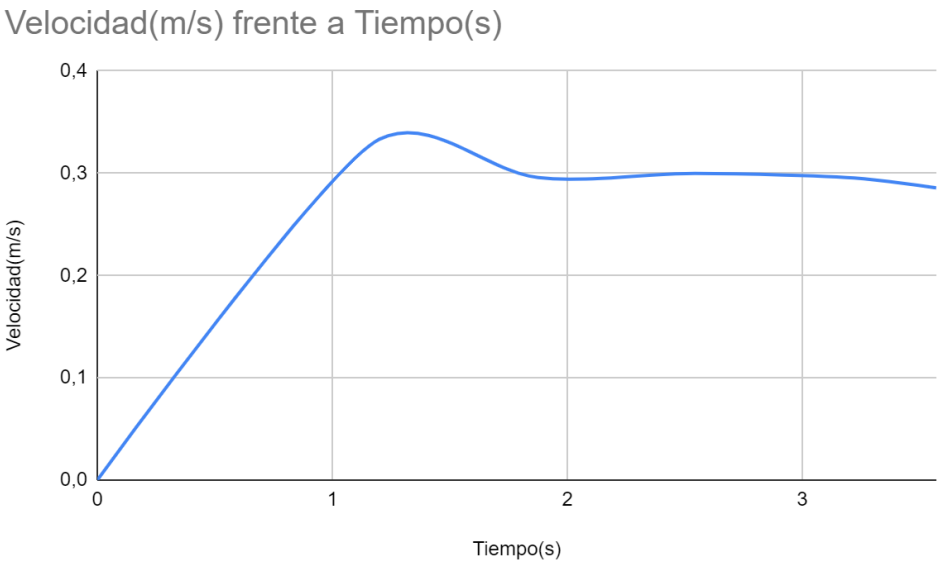

Fig 11.

Por lo que se observa es un sistema de segundo orden subamortiguado.

Luego procedemos a calcular

```
%% Análisis sistema según datos obtenidos en mediciones de velocidad
-----------------------------------------------------------------
yss=0.3;
ymax=0.33;
tp=1.2;

s=tf('s');
K=yss;
Mp=(ymax-yss)/yss;
psita=sqrt(log(Mp)^2/(pi^2+log(Mp)^2))
wn=pi/(tp*sqrt(1-psita^2))
G=tf(K*wn^2,[1 2*psita*wn wn^2])
step(G)
-----------------------------------------------------------------
```

Obteniendo como resultado:

```
psita = 0.5912
wn = 3.2459
Transfer function 'G' from input 'u1' to output ...

             3.161
 y1:  ---------------------
      s^2 + 3.838 s + 10.54
Continuous-time model.

polos =

  -1.9188 + 2.6180i
  -1.9188 - 2.6180i

ceros = [](0x1)
ganancia = 0.3000
```

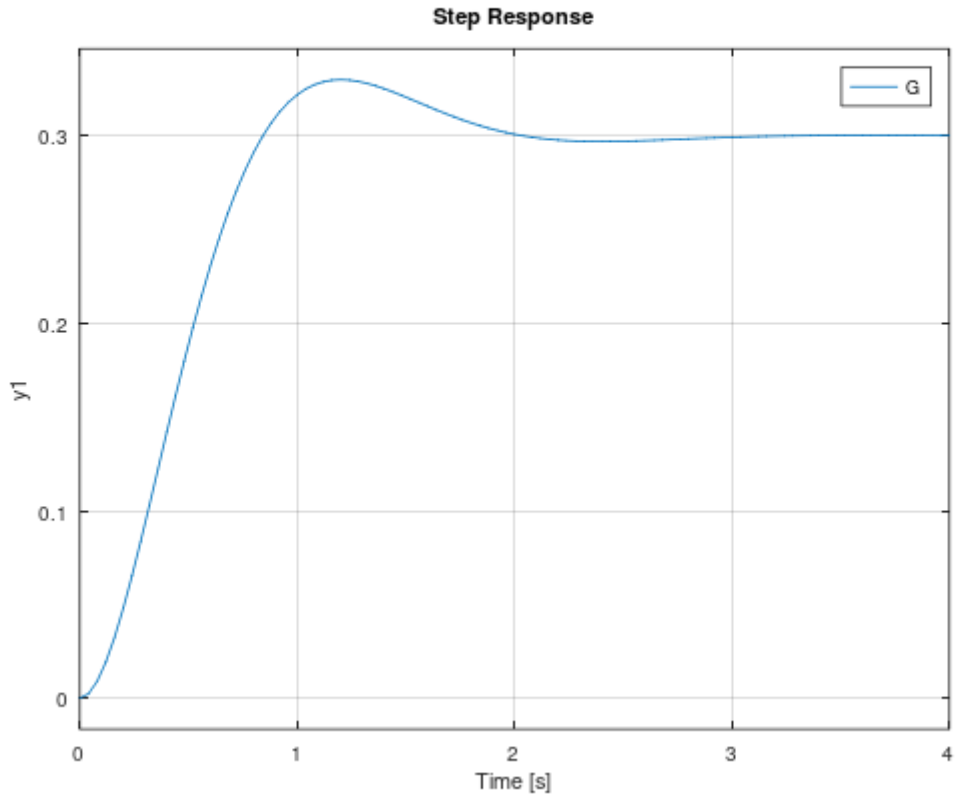

Fig 12.

Lecciones aprendidas

Se alcanzaron los objetivos planteados al comienzo del desarrollo del proyecto. Se puede ver que el proyecto tiene muchos aspectos a mejorar, como puede ser movilidad, energización, diseño pero debido al tiempo que teníamos para trabajar en el mismo, sumado a la carga horaria de las otras materias de este año, no se realizó un controlador PID tal como teníamos planeado ya que pusimos énfasis en el control inalámbrico y uso de módulos extras como en el acelerómetro. Se podría aplicar un controlador PID en el control de velocidad de nuestros motores por ejemplo para cuando doble nuestro auto gire de manera más progresiva y controlada. Además, la batería que iba en el mismo no era la más eficiente pero cumplía con su funcionamiento.

El uso de la IMU fue una experiencia nueva para todos los integrantes de nuestro grupo, el cual gracias a consideraciones del docente y comentarios de compañeros pudimos hacer uso de la misma a pesar de los inconvenientes.

Se podrían corregir bien los límites de los ángulos, de modo que el software se adapte bien al modelo de nuestro driver del motor el cual sabemos que tiene sus peculiaridades.

Imprevistos

- A la hora de querer implementar el controlador PID , no era lo óptimo aplicarlo en el sensor, a cambio de esto estudiamos y aplicamos lo denominado filtro Complementario o Kalman.
- Sensor IMU debe ser leído en el void loop del código sino genera problemas
- Medidas de v-t para trazar la curva, se debe tener en cuenta la longitud del auto-arduino.
- Inconvenientes en el tratado de ángulos y características del motor, lo que no nos permitió implementar giros más eficientes.

Conclusión

Como conclusión del proyecto podemos destacar que conseguimos poner en práctica los conocimientos aprendidos en las materias de sistemas de control 1 y 2, teoría de las comunicaciones, física e informática. Nos encontramos con varios desafíos, tuvimos que incorporar nuevos conocimientos como el funcionamiento y uso de acelerómetros, filtros kalman, programación en arduino y el uso de sus periféricos.

Esta tecnología tiene miles de aplicaciones y se podría seguir desarrollando como por ejemplo para controles de ascenso y descenso de vehículos eléctricos.

Anexo

Datasheet componentes utilizados:

Acelerómetro MPU6050:
https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

Transceptor NRF24L01:
https://pdf1.alldatasheet.com/datasheet-pdf/view/1489848/NORDIC/NRF24L01.html

Arduino Uno:
https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

Driver L298N:
https://pdf1.alldatasheet.es/datasheet-pdf/view/22440/STMICROELECTRONICS/L298N.html

Códigos en GitHub
https://github.com/FacuNRodriguez/HandCar.git